Apuntes U 2

Tema ***Estructuras de datos***

*"**Los hombres sabios aprenden con los errores que otros cometen; los tontos, con los propios**".*
Henry George Bohn

Definiciones

Estructura: La estructura (del latín structūra) es la disposición y orden de las partes dentro de un todo. También puede entenderse como un sistema de conceptos coherentes enlazados, cuyo objetivo es precisar la esencia del objeto de estudio.

Tipos de estructuras:

- ESTÁTICAS: donde su tamaño es fijo (como ser arrays, cadenas).
- DINÁMICAS: donde su tamaño es variable (como ser pilas, colas, listas).

- Lineal: aquellas en que el almacenamiento se hace en zonas contiguas.
- No Lineal: aquellas en los cuales el almacenamiento puede hacerse en cualquier zona, donde el concepto de "siguiente" es lógico y no físico

Abstracción: Una forma de resolver un problema es analizar las características y condiciones del mismo y de este análisis armar un modelo, el cual nos ayuda para llegar a una solución.

Una forma de resolver un problema es analizar las características y condiciones del mismo y de este análisis armar un modelo, el cual nos ayuda para llegar a una solución.

VENTAJA
Independizar el diseño del programa de la implementación especifica de los datos

Ese modelo es un concepto abstracto, el cual fue creado por nosotros para poder resumir el problema planteado, de una manera que nos resulta manejable.
Los pasos son por medio de acercamientos sucesivos al entorno real.
Partimos desde una visión de alto nivel y luego la vamos refinando añadiendo detalles y consideraciones particulares, hasta que obtenemos una descripción detallada o de "bajo nivel".
Una forma de resolver un problema es analizar las características y condiciones del mismo y de este análisis armar un modelo, el cual nos ayuda para llegar a una solución.

Ese modelo es un concepto abstracto, el cual fue creado por nosotros para poder resumir el problema planteado, de una manera que nos resulta manejable.
Los pasos son por medio de acercamientos sucesivos al entorno real.
Partimos desde una visión de alto nivel y luego la vamos refinando añadiendo detalles y consideraciones particulares, hasta que obtenemos una descripción detallada o de "bajo nivel".

Estructuras Estáticas

* Son aquellas en las que el tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse dicho tamaño durante la ejecución del programa.
Estas estructuras están implementadas en casi todos los lenguajes.
Ejemplos:
1.- Simples
    a) Boolean
    b) Char
    c) Integer
    d) Real

2.- Compuestas
    a) Arreglos
    b) Conjuntos
    c) Strings
    d) Registros

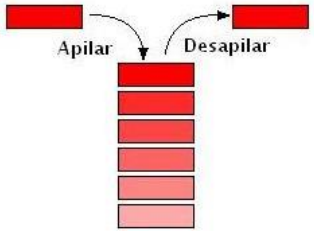
Existen estructuras como las ***pilas***, ***colas*** y ***listas*** que tienen una **implementación** estática y lineal.

Pilas o Stacks

- *Que es una Pila*

Una PILA es una lista lineal de elementos en la cual cada elemento sólo puede ser insertado o eliminado por un extremo denominado CIMA, es decir, los elementos se van a sacar de la pila en orden inverso al que se insertan (pila LIFO, Last In, First Out -ultimo en entrar primero en salir).

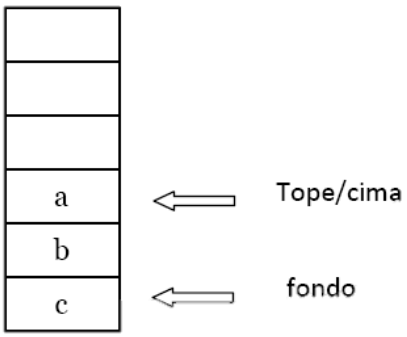
- *Operaciones básicas:*



• insertar: **Apilar**. Insertar un elemento en la cima.
• eliminar: **Desapilar**. Eliminar el elemento de la cima, en orden inverso a la inserción
• buscar: **Cima**. Devolver el elemento que se encuentra en la cima.

- *Otras operaciones:*
- • Determinar si la pila esta vacía
- • Determinar si la pila está llena

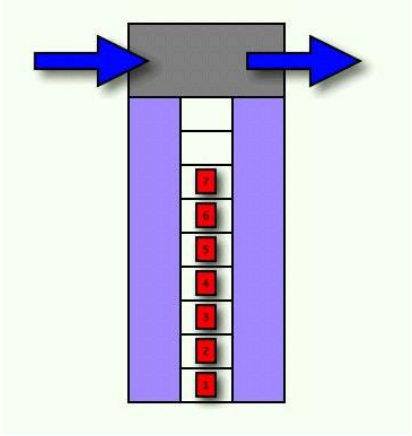
- *Representación*



Las características propias de la Pila nos permite utilizarlas en varias aplicaciones facilitándonos:
• Controlar, desde el Sistema Operativo, la ejecución de todas las ordenes de un archivo batch.
• Recordar al programa principal el punto de llamada de un subprograma y retornar al mismo cuando este termine.
• Formar cadenas de caracteres.
• Separar texto.
• Evaluar expresiones matemáticas.
• Deshacer la última operación realizada, por ejemplo, en un procesador de texto u otros programas similares.
Ahora bien, sobre la PILA podemos realizar las operaciones de:
- PUSH: poner o meter
- POP: sacar
Las pilas pueden ser representadas por medio de una lista unidireccional o por un array lineal.
Representaremos las pilas por medio de un array lineal.
Tendremos un puntero especial apuntando a la CIMA y una variable MAXPILA que determina el tamaño máximo de la pila.
Para controlar el OVER-UNDERFLOW hay que saber determinar el tamaño de la pila, porque si lo hacemos demasiado grande no habrá OVERFLOW pero tendremos una pérdida importante de memoria y viceversa.
Cada vez que un subprograma llama a otro, en la pila del sistema tiene que almacenarse la dirección del programa que llama, a la que se debe regresar cuando el subprograma llamado termina su ejecución.
La dirección de retorno es siempre la siguiente instrucción a la de la llamada.
El 1º en retornar será el último en efectuar la llamada y por eso utilizaremos una pila, dado la particularidad LIFO que mencionamos.

Las pilas y los lenguajes de programación

Las características propias de la Pila nos permite utilizarlas en varias aplicaciones facilitándonos:
• Controlar, desde el Sistema Operativo, la ejecución de todas las ordenes de un archivo batch.
• Recordar al programa principal el punto de llamada de un subprograma y retornar al mismo cuando este termine.
• Formar cadenas de caracteres.
• Separar texto.
• Evaluar expresiones matemáticas.
• Deshacer la última operación realizada, por ejemplo, en un procesador de texto u otros programas similares.



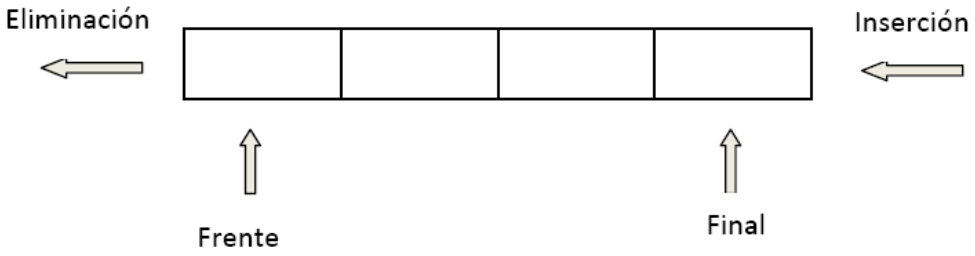
Observaciones:
Las pilas pueden ser representadas por medio de una lista unidireccional o por un array lineal.
Representaremos las pilas por medio de un array lineal.
Tendremos un puntero especial apuntando a la CIMA y una variable MAXPILA que determina el tamaño máximo de la pila.
Para controlar el OVER-UNDERFLOW hay que saber determinar el tamaño de la pila, porque si lo hacemos demasiado grande no habrá OVERFLOW pero tendremos una pérdida importante de memoria y viceversa.

Colas o Queues

Que es una cola
Es otro tipo de estructura de datos, cuyo acceso está restringido. La particularidad de una estructura de datos de cola es el hecho de que sólo podemos acceder al primer y al último elemento de la estructura. Así mismo, los elementos sólo se pueden eliminar por el principio y sólo se pueden añadir por el final de la cola.

Operaciones Básicas:
• Encolar (añadir, entrar, insertar): se añade un elemento a la cola. Se añade al final de esta.
• Desencolar (sacar, salir, eliminar): se elimina el elemento frontal de la cola, es decir, el • primer elemento que entró.
• Frente o primero (consultar, front): se devuelve el elemento frontal de la cola, es decir, el primer elemento que entró.

Representación



Otras Colas

Bicolas:
Son colas en donde los nodos se pueden añadir y quitar por ambos extremos; se les llama DEQUE (Double Ended QUEue). Para representar las bicolas lo podemos hacer con un array circular con Inicio y Fin que apunten a cada uno de los extremos.

Tipos de Bicolas:
• Bicolas de entrada restringida: Son aquellas donde la inserción sólo se hace por el final, aunque podemos eliminar al inicio o al final.
• Bicolas de salida restringida: Son aquellas donde sólo se elimina por el inicio, aunque se puede insertar al inicio y al final.

Otros Tipos
• Colas circulares: en las que el último elemento y el primero están unidos.
• Cola de prioridad: En ellas, los elementos se atienden en el orden indicado por una prioridad asociada a cada uno. Si varios elementos tienen la misma prioridad, se atenderán de modo convencional según la posición que ocupen.

Colas de Prioridad

Ejemplo: Cuando tenemos un sistema monoprocesador con problemas de paralelismo y concurrencia, podemos ver que la solución es que los procesos que quieran ejecutarse se vayan almacenando en una cola. Esto nos permite que, a medida que acaba un proceso en CPU, comience a ejecutarse el siguiente de la COLA.
Para solucionar esto, a cada proceso se le asigna una prioridad tal que se ejecutarán primero los de prioridad más alta. Inclusive se puede manejar la cantidad de procesos permitidos por prioridad e

inclusive asociar otras variantes a la prioridad, ejemplo tiempo de espero máximo permitido, tiempo de ejecución permitido, uso de memoria, etc.

Entonces, ¿que es una cola de prioridades?
Una cola de prioridades es un conjunto de elementos tal que cada uno se le ha asignado una prioridad de forma que el orden en el que los elementos son eliminados de la cola siguen el siguiente orden:
• Los elementos son procesados por orden de mayor prioridad
• Los elementos que tienen igual prioridad, según el orden en el que llegaron, es decir en forma convencional.

Representación:
- Por medio de una lista unidireccional
- Por múltiples colas

Estructura:
Si la representamos con una lista enlazada, tendrá la siguiente estructura:
- Un campo INFO
- Un campo enlace SIG; que apunta al siguiente elemento de la cola.
- Un campo NPR; que contiene la prioridad de ese elemento.

Tipos
• Con ordenamiento descendente
• Con ordenamiento ascendente

Colas de prioridad ascendente
En las colas de prioridad ascendente (CPA) se pueden insertar elementos en forma arbitraria y solamente se puede remover el elemento de menor prioridad.
Las operaciones que podemos hacer:
- Insert(CPA,x), inserta el elemento x en la cola CPA.
- x=minRemove(CPA) asigna a x el valor del elemento menor (de su prioridad) y se remueve de la cola.

Colas de prioridad descendente
En las colas de prioridad descendente (CPD) es similar, pero sólo permite la supresión del elemento más grande; las operaciones que podemos realizar con este tipo de cola son:
• insert(CPD,x)
• x=maxRemove(CPD)

Con los 2 tipos de colas
La operación empty , la operación insert y la operación borrar, que sólo se aplica si no esta vacía.

Observaciones:
Los elementos de la cola de prioridad no necesitan ser números o caracteres para que puedan compararse directamente, de hecho pueden ser estructuras complejas ordenadas en uno o varios campos.
En las colas de prioridad se pueden sacar los elementos que no están en el primer sitio del extremo donde salen los elementos, dado que el elemento a retirar puede estar en cualquier parte del arreglo.

Cuando se requiere eliminar un dato de una cola de prioridad se necesita verificar cada uno de los elementos almacenados para saber cuál es el menor (o el mayor). Esto trae algunos problemas, el principal es que el tiempo necesario para eliminar un elemento puede crecer tanto como elementos tenga la cola.

Para resolver este problema hay varias soluciones:
Se coloca una marca de "vacío" en la casilla de un elemento suprimido. Este enfoque realmente no es muy bueno, porque de cualquier modo se accesan los elementos para saber si es una localidad vacía o no lo es. Por otro lado, cuando se remueven elementos, se van creando lugares vacíos y después es necesario hacer una compactación, reubicando los elementos en el frente de la cola.

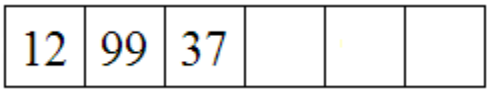
Cada supresión puede compactar el arreglo, cambiando los elementos después del elemento eliminado en una posición y después decrementando en 1. La inserción no cambia. En promedio, se cambian la mitad de los elementos de una cola de prioridad para cada supresión, por lo que esta operación no es eficiente.

Listas (estáticas)

Que es una Lista
Es un conjunto de elementos de un tipo dado que se encuentran ordenados y pueden variar en número. Los elementos en una lista lineal se almacenan en la memoria principal de una computadora en posiciones sucesivas de memoria

Operaciones (en forma genérica):
- Agregar; borrar.
- Recorrer la lista para localizar algún elemento
- Clasificar los elementos en orden ascendente o descendente
- Al conocer la cantidad de nodos y sus ubicaciones, casi no hay restricciones en lo que se puede hacer

Representación



Notas:
Si hablamos de listas en general, no podemos especificar sus características y nos arriesgamos a caracterizarla como dinámica.
Pero si decimos que es una Lista estática, podemos categorizarla como estática y lineal, siendo su implementación mas común en un arreglo unidimensional.

Estructuras Dinámicas

* No tienen las limitaciones o restricciones en el tamaño de memoria ocupada que son propias de las estructuras estáticas.
Mediante el uso de un tipo de datos especifico, denominado puntero (por razones técnicas podremos llamarlo referencia o dato referencial), es posible construir estructuras de datos dinámicas que no poseen nativamente los lenguajes.
Se caracterizan también por el hecho de no saber en que lugar de la memoria se alojaran.

NOTA: Mas allá de sus implementaciones estáticas, las Pilas, Colas y Listas casi siempre se implementan como dinámicas haciendo uso de estructuras con datos y datos referenciales.
Lenguajes como C++ y Pascal tiene un tipo de datos espacial llamado puntero. En Java al no tener ese tipo de datos utilizamos las referencias a los objetos creados (datos referenciales).

¿Que es una referencia?

Las referencias en Java no son punteros ni referencias como en C++. Las referencias en Java son identificadores de instancias de las clases Java. Una referencia dirige la atención a un objeto de un tipo específico. No tenemos por qué saber cómo lo hace ni necesitamos saber qué hace ni, por supuesto, su implementación.

Listas enlazadas - Estructura dinámica

¿ Que es una Lista enlazada o lista ligada?
Una lista enlazada consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior. El principal beneficio de las listas enlazadas respecto a los vectores convencionales es que el orden de los elementos

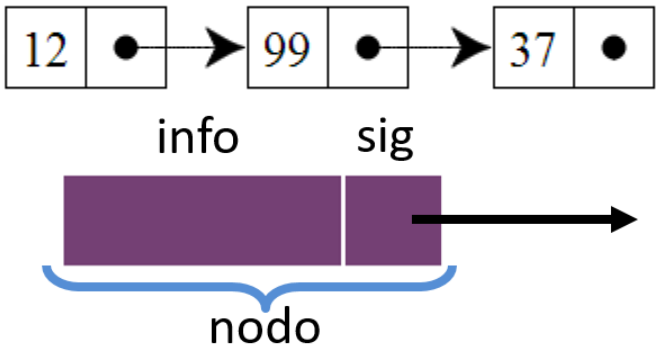
enlazados puede ser diferente al orden de almacenamiento en la memoria o el disco, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.

Una lista enlazada es un tipo de dato autorreferenciado porque contienen un puntero o enlace (en inglés link, del mismo significado) a otro dato del mismo tipo. Las listas enlazadas permiten inserciones y eliminación de nodos en cualquier punto de la lista en tiempo constante (suponiendo que dicho punto está previamente identificado o localizado), pero no permiten un acceso aleatorio. Existen diferentes tipos de listas enlazadas: listas enlazadas simples, listas doblemente enlazadas, listas enlazadas circulares y listas enlazadas doblemente circulares.

Las listas enlazadas pueden ser implementadas en muchos lenguajes. Lenguajes tales como Lisp y Scheme tiene estructuras de datos ya construidas, junto con operaciones para acceder a las listas enlazadas. Lenguajes imperativos u orientados a objetos tales como C o C++ y Java, respectivamente, disponen de referencias para crear listas enlazadas.

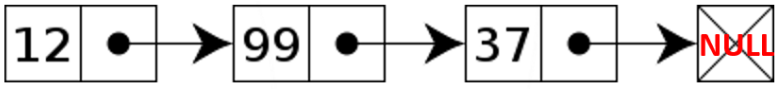
Operaciones:
• Crear: se crea la cola vacía.
• Agregar (añadir, entrar, insertar): se añade un elemento a la lista.
• Borrar (sacar, salir, eliminar): se elimina el elemento de la lista.
• Buscar (consultar, etc): Busca un nodo, comenzando desde el primero y siguendo las referencias.

Representación

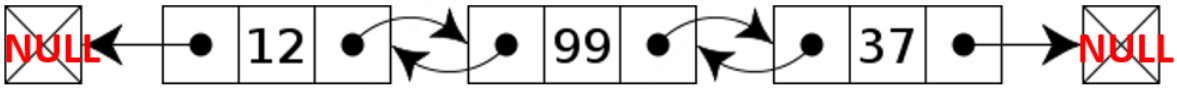


Tipos de Listas enlazadas

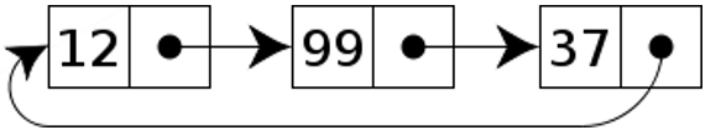
Listas simples enlazadas
Tiene un enlace por nodo. Este enlace apunta al siguiente nodo en la lista, o al valor Nulo o lista Vacía, si es el último nodo.



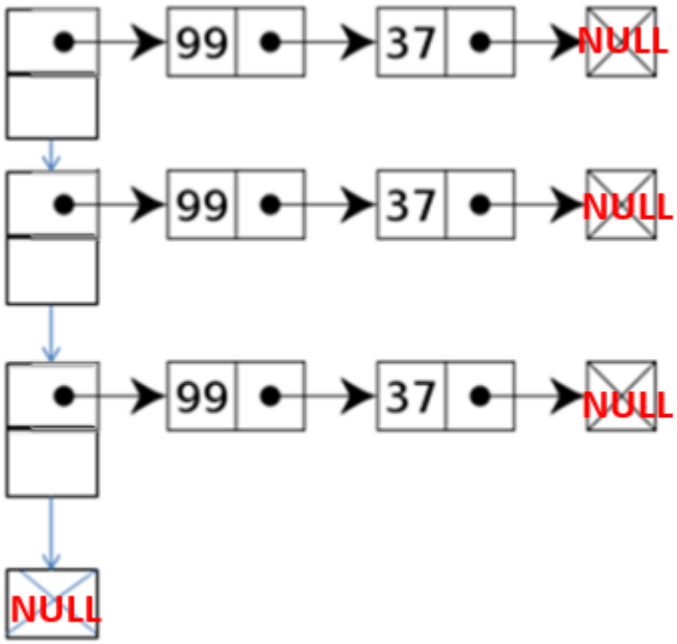
Listas doblemente enlazadas:
Cada nodo tiene dos enlaces: uno apunta al nodo anterior, o apunta al valor Nulo o la lista vacía si es el primer nodo; y otro que apunta al siguiente nodo, o apunta al valor Nulo o la lista vacía si es el último nodo.



Listas enlazadas circulares
En una lista enlazada circular, el primer y el último nodo están unidos. Para recorrer una lista enlazada circular podemos empezar por cualquier nodo y seguir la lista en cualquier dirección hasta que se regrese al nodo original



Listas de listas
El campo de datos de un nodo puede ser otra lista enlazada.



Operaciones de listas

Recorrido
Consiste en visitar cada uno de los nodos que forman la lista.

Inserción
Consiste en agregar un nuevo nodo a la lista (al inicio, antes o después de cierto nodo o Insertar un nodo al final)

Borrado
Consiste en quitar un nodo de la lista, redefiniendo las ligas que correspondan.(el primer nodo, el último nodo, un nodo con cierta información, etc.)

Búsqueda
Consiste en visitar cada uno de los nodos hasta encontrar el elemento dado.

Casos de inserción
1. -Que el nodo insertar pase a ser el primero de la lista. Entonces tendré que variar la variable comienzo.
2. -Que el nodo insertar pase a ser el último de la lista por lo que SIG tendría que pasarlo a NULL
3. -Insertar un nodo a partir de otro concreto.

Pasos a seguir:
A) Estudiar si existe espacio libre en la lista. Si no hay entonces se produciría un OVERFLOW y no puedo hacer la inserción.
B) Crear el nuevo nodo y actualizar los punteros necesarios en esta lista.()
C) Rellenar el nuevo nodo con la información que queremos insertar en él.
D) Reajustar los campos de enlace que sean necesarios en la lista con la que estamos trabajando.

Inserción de un nodo en una lista ordenada
A. -Ver si hay espacio
B. -Si la lista está vacía o si el elemento a insertar es menor que el primero de la lista aplico el algoritmo 1.
C. -Si no se cumplen las condiciones anteriores tendré que localizar la posición LUG que le corresponde al nodo e insertarlo allí con el algoritmo 3 o 2 (si es el último).

Casos de Borrado
1. - Que el nodo a borrar sea el primero de la lista.
2. - Borrar un nodo conocido su predecesor.
   Caso particular: borrar un nodo que está el final de la lista
3. - Borrado de un nodo que contiene un determinado elemento de información.

Consideraciones

Nodos de referencia o nodos centinelas: A veces las listas enlazadas tienen un nodo centinela (también llamado falso nodo o nodo ficticio) al principio o al final de la lista, el cual no es usado para guardar datos. Su propósito es simplificar o agilizar algunas operaciones, asegurando que cualquier nodo tiene otro anterior o posterior, y que toda la lista (incluso alguna que no contenga datos) siempre tenga un "primer y último" nodo. (algunos lo llaman cabecera y tierra)

Overflow y Underflow (en estructuras dinámicas)

Una lista sufre un OVERFLOW cuando se intenta insertar un elemento en dicha lista estando la lista de espacio disponible vacía, es decir, cuando no hay más memoria libre.

Una lista sufre UNDERFLOW cuando intentamos borrar un elemento de la lista estando ésta vacía.

Estas 2 situaciones habrá que controlarlas siempre que hagamos una inserción y siempre que hagamos un borrado.